

Finite Domain Constraint Programming Systems

Mats Carlsson

SICS

Uppsala, Sweden

matsc@sics.se



Christian Schulte

IMIT, KTH

Stockholm, Sweden

schulte@imit.kth.se



KUNGL.
TEKNISKA
HÖGSKOLAN



What Is This Tutorial About?



- Focus is on
 - *Services systems provide*
 - *Implementation of these systems*
- No detailed description of one system
 - *Common techniques, approaches, challenges*
 - *No intention to do complete survey*

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

2

Finite Domain Constraint Programming Systems



- Offer *reusable* software components for
 - constraint propagation
 - combining constraints (combinators)
 - search
 - branching (labeling)
 - exploration (for example: depth-first, LDS, ...)
 - user extensions
- Services provided
 - environment for integrating components
 - libraries of commonly used components

[Henz & Müller 00]

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

3

Systems Discussed



- Prolog-based systems
 - SICStus Prolog, Eclipse Prolog, GNU Prolog, CHIP,...
- Libraries
 - ILOG Solver (C++) and JSolver (Java), Choco (Claire), Figaro (C++), Facile (Ocaml), CHIP Library (C++)
- Specialized languages
 - Claire, Oz

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

4

Outline of Tutorial

- Constraint propagation
 - example [Christian]
 - model [Christian]
 - implementation [Mats]
 - optimizations [Mats]
- Search [Christian]
- Combinators [Christian]
- Trends & Challenges [Mats]
- References

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

5

Constraint Propagation

Example

Example

$$x + y = 9$$

$$2x + 4y = 24$$

x	0	1	2	3	4	5	6	7	8	9
y	0	1	2	3	4	5	6	7	8	9

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

7

Example

$$x + y = 9$$

$$2x + 4y = 24$$

x	0	1	2	3	4	5	6	7	8	9
y	0	1	2	3	4	5	6	7	8	9

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

8

Example

$$x + y = 9$$

$$2x + 4y = 24$$

x	0	1	2	3	4	5	6	7	8	9
y	0	1	2	3	4	5	6	7	8	9

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

9

Example

$$x + y = 9$$

$$2x + 4y = 24$$

x	0	1	2	3	4	5	6	7	8	9
y	0	1	2	3	4	5	6	7	8	9

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

10

Example

$$x + y = 9$$

$$2x + 4y = 24$$

x	0	1	2	3	4	5	6	7	8	9
y	0	1	2	3	4	5	6	7	8	9

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

11

Example

$$x + y = 9$$

$$2x + 4y = 24$$

x	0	1	2	3	4	5	6	7	8	9
y	0	1	2	3	4	5	6	7	8	9

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

12

Example

$$x + y = 9$$

$$2x + 4y = 24$$

x	0	1	2	3	4	5	6	7	8	9
y	0	1	2	3	4	5	6	7	8	9

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

13

Constraint Propagation

Model

Constraint Propagation

- Variables
 - feature variable domain (finite set of integers)
- Propagators
 - implement constraints
- Propagation loop
 - execute propagators until simultaneous fixpoint

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

15

Propagator

- Propagator p is procedure
 - implements constraint $\text{con}(p)$
its semantics (set of tuples)
 - computes on set of variables $\text{var}(p)$
- Execution of propagator p
 - narrows domains of variables in $\text{var}(p)$
 - signals failure
 - signals entailment [discussed later]

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

16

Classes of Constraints

- Basic constraints
 - Constraints for which the solver is complete
 - $x \in D$, $x = v$, $x = y$ (variable aliasing)
- Primitive constraints (need propagators)
 - Non-decomposable constraints
 - $x < y$, $x \neq y$, $x + y = z$, $x * y = z$, ...
- Global constraints (need propagators)
 - Subsume a set of basic or primitive constraints, usually providing stronger consistency

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

17

Reification

- Control constraint by control variable $b \in \{0,1\}$
$$c \Leftrightarrow b=1$$
 - also require propagator
- A.k.a.: metaconstraints

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

18

Propagators Are Intensional

- Propagators implement narrowing
 - also: filtering, propagation, domain reduction
- No extensional representation of $\text{con}(p)$
 - impractical in most cases (space)
- Extensional representation of constraint
 - can be provided by special propagator
 - often: "element" constraint, "relation" constraint, ...

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

19

Implementing Propagators

- Implementation uses operations on variables
 - reading domain information
 - narrowing domains
- Variables are the only communication channels between propagators
- More detail later

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

20

Propagator Properties

- Propagator p is
 - correct: no solution of $\text{con}(p)$ is removed
 - assignment complete: failure at latest for assignments
 - compatibility with search
- Propagator p is
 - contracting: variable domains are narrowed
 - monotonic: application to smaller domains will result in smaller domains than application to larger domains
 - may be idempotent: [discussed later]

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

21

Propagation Loop

- Largest simultaneous fixpoint of propagators
 - fixpoint: propagators cannot narrow any further
 - largest: no solutions lost
- Guaranteed
 - termination: domains finite
propagators contracting
 - largest fixpoint: propagators monotonic

Detailed study with proofs: [Apt 00]

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

22

Fix and Runnable Propagators

- Propagator is either
 - fix: has reached fixpoint
 - runnable: not known to have reached fixpoint
- Propagation loop maintains propagator sets
 - all propagators $Prop$
 - runnable propagators Run
 - initially $Run := Prop$

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

23

Sketch of Propagation Loop

```

while ( $Run \neq \emptyset$ ) {
  pick and remove  $p$  from  $Run$ ;
  execute  $p$ ;
   $ModVar := \{ x \mid x \text{ modified by } p \}$ ;
   $DepProp := \{ q \mid x \in \text{var}(q), x \in ModVar \}$ ;
   $Run := \text{join}(DepProp, Run)$ ;
}

```

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

24

Sketch of Propagation Loop

```
while ( $Run \neq \emptyset$ ) {  
    pick and remove  $p$  from  $Run$ ;  
    execute  $p$ ;  
     $ModVar := \{ x \mid x \text{ modified by } p \}$ ;  
     $DepProp := \{ q \mid x \in \text{var}(q), x \in ModVar \}$ ;  
     $Run := \text{join}(DepProp, Run)$ ;  
}
```

Loop invariant: $p \text{ is fix} \Leftrightarrow p \in (Prop-Run)$

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

25

Sketch of Propagation Loop

```
while ( $Run \neq \emptyset$ ) {  
    pick and remove  $p$  from  $Run$ ;  
    execute  $p$ ;  
     $ModVar := \{ x \mid x \text{ modified by } p \}$ ;  
     $DepProp := \{ q \mid x \in \text{var}(q), x \in ModVar \}$ ;  
     $Run := \text{join}(DepProp, Run)$ ;  
}
```

Termination ($Run=\emptyset$): $p \text{ is fix} \Leftrightarrow p \in Prop$

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

26

Sketch of Propagation Loop

```
while ( $Run \neq \emptyset$ ) {  
    pick and remove  $p$  from  $Run$ ;  
    execute  $p$ ;  
     $ModVar := \{ x \mid x \text{ modified by } p \}$ ;  
     $DepProp := \{ q \mid x \in \text{var}(q), x \in ModVar \}$ ;  
     $Run := \text{join}(DepProp, Run)$ ;  
}
```

Ignored: failure (signaled by p)

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

27

Consistency Level Computed

- Model is generic
- Consistency level defined by each individual propagator
 - accurate way of characterization [Maher 02]
- Supports many different consistency levels
 - propagator for domain-consistent alldifferent
 - propagator for bound-consistent alldifferent
 - propagator for value-consistent alldifferent

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

28

Major Design Decisions

- Implementing *Run* (that is, pick and join)
 - queue: first in – first out
 - stack: last in – first out
 - priority queue
- Implementing *ModVar* and *DepProp*
 - variable-centered representation

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

29

Implementing *ModVar* and *DepProp*

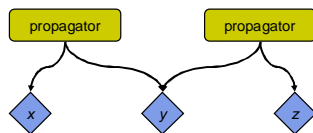
- Variable-centered approach
 - each variable x knows dependent propagators
 - typically organized as list (*suspension list*)
 - propagator p included in list of $x \Leftrightarrow x \in \text{var}(p)$
- Upon propagator creation
 - propagator subscribes to its variables
 - becomes runnable

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

30

Propagators \Rightarrow Variables



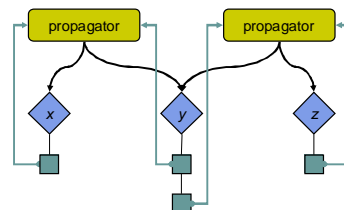
- **Propagators** know their **variables**
 - to perform domain modifications
 - passed as parameters to propagator creation

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

31

Variables \Rightarrow Propagators



- **Variables** know dependent **propagators**
 - to perform efficient computation of dependent propagators
 - implemented by *suspension lists*

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

32

Modifying a Variable

- Traverse suspension list
 - add propagators to *Run*
- Optimization
 - mark runnable propagators
 - that is: propagators already in *Run*
- Multiple variable modification by propagator
 - explicitly maintain *ModVar* (as in model)
 - only after propagator execution: process *ModVar*
 - suspension list traversed only once per variable

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

33

Propagation Events

- Use knowledge on propagator to avoid its redundant execution
 - redundant: application to fixpoint
- Example: bound-consistent linear equality
 - need to execute, if bound of variable changes
 - no need to execute, if inner value of variable removed
- Suspension list: $\langle \text{propagator}, \text{event} \rangle$
 - event describes relevant domain modifications
 - implementation: lists per event, single list of pairs
 - events: VALUE, BOUND, DOMAIN

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

34

Idempotent Propagators

- Idempotent propagator
 - always computes fixpoint
- Propagation loop perspective
 - no need to include in *Run*
 - more efficient: saves one invocation of propagator
- Propagator perspective
 - must compute fixpoint itself
 - more efficient: specific method for computing fixpoint
 - might be more challenging

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

35

Propagator Entailment

- Propagator will never contribute anything
 - fixpoint property preserved by narrowing
- Delete propagator, if entailment detected
 - remove from suspension-list, or
 - mark as dead, delegate removal to garbage collection
- Similar to consistency, different entailment levels
 - semantically relevant in concurrent constraint programming

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

36

Summary of Model

- Variables
 - domain
 - suspension list: (event, propagator)
- Propagators
 - intensional, correct, contracting, monotone, define consistency level, ...
 - know variables for narrowing
- Propagation loop
 - computes largest simultaneous fixpoint

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

37

Constraint Propagation

Implementation

Propagation Queues

- Contents
 - Events, variables, or propagators
- Scheduling Policy
 - LIFO makes sense for "important" events
 - FIFO – fair scheduling, no starvation
 - Compare LIFO and FIFO for:
 - $x > y, y > x, x \geq 100t, y \leq t, \{x, y, t\} \in 1..1000$
- Structure
 - Flat or layered

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

39

Layered Propagator Queue

- CHOCO's 8 priority levels
 - VALUE event queue: LIFO
 - BOUND event queue: FIFO
 - DOMAIN event queue: FIFO
 - P's for extensionally defined constraints (AC-4)
 - $O(N)$ propagators
 - $O(< N^2)$ propagators
 - $O(N^2)$ propagators
 - $O(> N^2)$ propagators

[Laburthe 00]

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

40

Stateful Propagators

- Prerequisite for incremental algorithms
 - Can bring down complexity by an “order”
- Arguments checked initially only
- Prolog level state not enough
- State can be used for:
 - Gradually ignoring ground variables
 - Data structures for the filtering algorithm
 - Memory of variables’ min/max/domain
 - Local trailing for backtracking

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

41

What Does a Propagator Know?

- Course-grained information
 - Something has changed (SICStus, Mozart)
- Medium-grained information
 - Variables v_3, v_7, v_{11} have changed (CHIP)
- Fine-grained information (ILOG)

Variable	Old domain	New domain	Delta
v_1	1..5	2..4	{1,5}
- Stateful propagators can figure out what changed

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

42

What Does a Propagator Need?

- A good filtering algorithm
 - An algorithm library might come in handy
 - Shortest-path, bipartite matching, max flow, min-cost flow, profiles, strongly connected components, ...
- ADT: finite domain
- ADT: domain variable
- Host language services
- Solver kernel interface
 - True/false/suspend, replace_by, I_am_not_idempotent, ...
- State

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

43

ADT: Finite Domain

- Representation
 - List of intervals (ECLiPSe, SICStus)
 - Bounds + bit array (CHIP, GNU Prolog, Mozart)
 - Bounds + list of holes
 - Interval trees
 - Multiple, adaptive (CHOCO, Mozart, ...)
- Operations
 - Set operations
 - Constructors, iterators
 - Complexity depends on representation

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

44

ADT: Domain Variable

- Representation
 - Logic variable + attributes for domain and suspensions, or
 - Class instance
- Operations
 - Access min, max, domain
 - Adjust min, max, domain; remove values
 - Raise events
 - Attach/detach suspensions

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

45

Variable Aliasing ($x = y$)

- Only an issue in logic-based languages
- Normalization:
 - Merge suspension lists
 - Intersect domains
 - Raise events
 - IF $\text{con}(p)$ mentions both x and y THEN
 - p may no longer be idempotent
 - p can make more inferences, e.g.:
 $\text{xor}(x,y,z), x=y \Rightarrow z=0$

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

46

Host Language Services (Generic)

- Memory management
 - Allocation: objects, states
 - Garbage collections: term refs in states and queues
 - Copying
- Trailing
 - Coarse or fine
 - Semantic trailing for self-destruct on backtracking
- Resume/suspend mechanism
 - Full coroutining, multithreading etc. not needed

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

47

Host Language Services (Prolog-Based)

- Predicate/function type extensions
 - Constraints must be callable like predicates
 - WAM support for indexicals (coming slides)
- Attributed variables [Holzbaur 92] [Le Huitouze 90]
 - Domains
 - Suspensions
 - Unification hook
- Mutable terms
 - Coarse trailing

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

48

Implementation Choices

- In host language
 - Prolog (ECLiPSe, SICStus)
 - C++ (ILOG Solver, Figaro, CHIP Library)
 - Claire (CHOCO)
 - Java (ILOG JSolver)
- C/C++
 - For predefined constraints (ECLiPSe, SICStus)
 - For predefined + user-defined constraints (Mozart)
- Indexicals
 - For "pseudo primitives" (GNU Prolog, SICStus)

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

49

Indexicals

- Given $C(X_1, \dots, X_n)$, for each i , provide a rule $X_i \text{ in } R_i$ which computes the feasible values of X_i
 - Example: $X = Y + C$, arc-consistent version


```
eqcd(X, Y, C) +:
    X in dom(Y)+C,
    Y in dom(X)-C.
```
 - Example: $X = Y + C$, bound-consistent version


```
eqcd(X, Y, C) +:
    X in min(Y)+C..max(Y)+C,
    Y in min(X)-C..max(X)-C.
```

[Van Hentenryck & Deville & Saraswat 92]

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

50

Indexicals: Syntax of $X \text{ in } R$

Range expressions

$R ::= T..T \mid R \setminus R \mid R \vee R \mid R \mid R+T \mid R-T \mid R \bmod T \mid \{T, \dots, T\} \mid \text{dom}(X)$

Term expressions

$T ::= T+T \mid T-T \mid T/>T \mid T</T \mid T \bmod T \mid \min(X) \mid \max(X) \mid X \mid \text{integer} \mid \text{inf} \mid \text{sup}$

Monotone indexicals for propagation

Anti-monotone indexicals for entailment check

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

51

Indexicals for Reification

Example: $X = Y + C$, SICStus syntax

?- eqcd(X, Y, 5) #<=> B.

```
eqcd(X, Y, C) +:      % propagation
    X in dom(Y)+C, Y in dom(X)-C.
eqcd(X, Y, C) -:      % converse propagation
    X in \{Y+C}, Y in \{X-C}.
eqcd(X, Y, C) +?      % entailment check
    X in {Y+C}.
eqcd(X, Y, C) -?      % disentanglement check
    X in \dom(Y)+C.
```

[Carlsson & Ottosson & Carlson 97]

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

52

Indexicals: Implementation

- Compiled to (bytecode, symbol table)
- Syntax intercepted by *term expansion*
- Executed by a simple stack-based VM
- `eqcd/3` gets defined as a Prolog predicate
 - The WAM escapes to a solver entrypoint

[Carlsson & Ottosson & Carlson 97]

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

53

Indexicals: Pros and Cons

- Efficiency: witness GNU Prolog
- A RISC approach to constraint solving
- A VM for propagators
- A language for fine-tuned propagation in a general framework
- Can detect entailment as well as propagate
- Drawbacks
 - Pseudo-primitives only (no global constraints)
 - N propagators needed for 1 constraint

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

54

An Entity-Relationship Model (CHOCO, Figaro, ILOG)

- Objects
 - Problems
 - Variables
 - Domains
 - Constraints
- Relationships
 - Links between constraints and variables

(*constraint, variable, event*)

[Puget 94][Puget & Leconte 95] [Laborthe 00]

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

55

Constraint Objects

- State in private data
- Virtual methods for (some of):
 - Posting
 - Propagation
 - Entailment/disentailment test
 - Reification
 - Profiler and visualizer services
 - Memory manager services

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

56

The ILOG Global Constraint API (Simplified)



```
class ClassName : public IlcConstraint {
public:
    ClassName(IloSolver solver, Args);
    ~ClassName(void);
    virtual void post(void);
    virtual void propagate(void);
    virtual IlcBool isViolated(void) const;
};
```

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

57

The Mozart Global Constraint API (Simplified)



```
class ClassName : public OZ_Propagator {
public:
    ClassName(OZ_Term Args);
    virtual OZ_Return propagate(void);
    virtual size_t sizeOf(void);
    virtual void gCollect(void);
    virtual void sClone(void);
    virtual OZ_Term getParameters(void) const;
};
```

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

58

A Global Constraint API for Prolog (SICStus)



- `fd_global(+C, +S, +V)`
 - Posts a global constraint *C* with initial state *S*, suspended according to *V*, which is a list of `dom(X)`, `min(X)`, `max(X)`, `minmax(X)`, `val(X)`
- `dispatch_global(+C, +S0, -S, -A)`
 - user defined
 - Entrypoint to the propagator of constraint *C* with state *S0*, producing a new state *S* and kernel requests *A* (*true/false/suspend*, events)
- ADTs for domains and domain variables
- Control is implicit

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

59

Constraint Propagation

Optimizations



Rewriting

- Generic \rightarrow special
 $3x + y - z = 0, x = 0$
 \rightarrow
 $y = z$ (variable aliasing)
- Gradual decomposition
 $\text{alldiff}([T, U, V, X, Y, Z]), [T, U, V] \text{ in } 1..3, [X, Y, Z] \text{ in } 4..6$
 \rightarrow
 $\text{alldiff}([T, U, V]), \text{alldiff}([X, Y, Z])$

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

61

Avoiding Useless Work

- Idempotent p should be immune to events raised by p
 - Kernel may or may not assume idempotence
- An entailed p should never be resumed
 - It can even be detached (undoably) from $\text{var}(p)$
- IF time of latest event < time of latest resumption THEN don't resume p
 - Event queues require timestamps
- Indexicals linked to the same constraint should (sometimes) be immune to each other's events

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

62

More Optimizations

- Different event types
 - VALUE > BOUND > DOMAIN
 - p is suspended on a set of (v, event)
 - Demons vs. propagators
- Scheduling policies
 - Poorly understood
 - Complexity-based priority queues make sense
 - Always bear the worst case in mind

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

63

Search

Branching and Exploration

- Branching: defines shape of search tree
 - labeling, branching, distribution, ...
 - often based on heuristics
- Exploration: explore nodes of search tree
 - often fixed to be depth-first
 - many aspects
 - optimization (branch-and-bound)
 - development tools (Oz Explorer)
 - parallelism (ILOG Solver, Oz)

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

65

Branching

- Requires synchronization on fixpoint
 - for implementing dynamic variable orderings
 - by construction: Prolog, ILOG Solver, ...
 - explicit synchronization in concurrent setup: Oz
- Programmed
 - from builtin-search: Prolog-based
 - special (language) constructs: ILOG Solver, Oz
- Typically, rich library available

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

66

Exploration

- All systems support
 - search for first solution
 - search for some/all solutions
 - search for best solution
- Most systems support
 - LDS and some variants

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

67

Exploration Strategy

- Often fixed to be depth-first
- Sometimes can be programmed
 - Oz: spaces ("nodes") as ADT for exploration
 - exploration programmed from operations
 - for example: copy node in search tree
access solution
 - ILOG Solver: control exploration by limits and priorities
 - limit cut-off branches
 - priorities which node to explore next

[Schulte 97] [Perron 99]

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

68

Infrastructure for Exploration

- State restauration
 - backtrack to a *previous* state
- Approaches
 - trailing: recording and undoing changes
 - copying: put complete state aside
 - recomputation: recompute state on need
- By far dominating approach: trailing

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

69

Trailing

- Trailing stores undo and redo information
 - interleaved with constraint propagation
 - uses trail data structure
 - update: put (location, content)
 - undo: write location \leftarrow content
 - every choice point: put mark or record top of trail
- Requires
 - all updates trail-aware
 - for example: domain change, change of suspension list, ...

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

70

Time Stamping

- Problem: multiple change of same location
 - for example: multiple narrowing of domain
 - only original value needs restauration
 - intermediate values not needed
- Solution: local time stamp on modified entity
 - new choice point increase global time stamp
 - upon modification trail, if local stamp earlier
 - update local stamp

[Aggoun & Beldiceanu 90] [Aggoun & Beldiceanu 91]

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

71

Multiple Value Trail

- Modifying n successive locations
 - record start, number (n) and n locations on trail
 - instead of $2n$ individual entries

[Aggoun & Beldiceanu 90] [Aggoun & Beldiceanu 91]

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

72

Copying And Recomputation

- Copying
 - operations ignorant of state restoration
 - support for concurrency and parallelism
 - alone infeasible: excessive memory requirements
- Hybrid strategies: copying and recomputation
 - adaptive: create copy on demand to speed up future recomputation
 - batch: speed up recomputation by avoiding repeated fixpoint computation
 - competitive with trailing

[Schulte 99] [Choi & Henz & Ng 01] [Schulte 02]

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

73

Combinators

Combinators

- Reification-based combination
 - reified constraints
 - propositional combination
- Propagation-preserving approaches
- Constructive disjunction

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

75

Reification

- Use control variable $b \in \{0, 1\}$
 $c \Leftrightarrow b=1$
- Propagate
 - c entailed \Rightarrow propagate $b=1$
 - $\neg c$ entailed \Rightarrow propagate $b=0$
 - $b=1$ entailed \Rightarrow propagate c
 - $b=0$ entailed \Rightarrow propagate $\neg c$ (might be difficult)

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

76

Propositional Combination

- Constraints for combining reified constraints
 - constraints as connectives over 0/1 variables
- Combine $(c_1 \wedge c_2) \vee c_3$
 - reify each c_i to 0/1 variable
 - use constraints on 0/1 variables
- Problem: not propagation-preserving
 - no propagation between c_1 and c_2
 - in $c_1 \wedge c_2$, both c_1 and c_2 propagate individually

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

77

Compositional Approaches

- Use language-based primitives for implementing combinators
 - encapsulated propagation
 - generalization of ccp-paradigm
 - pioneered by AKL [Haridi & Janson 90]
 - generalized to programming abstraction [Schulte 02]
- Advantages and disadvantages
 - expressive and propagation preserving
 - implementation complex and less efficient than reification
 - unclear how to provide in language-independent setting

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

78

Constructive Disjunction

- Idea: make assumptions and generalize
 - propagate locally in each branch of disjunction
 - lift out common information on domains from branches
- Well researched/published idea
 - cc(FD) [Van Hentenryck & Saraswat & Deville 95]
 - many other papers, for example [Codognet & Codognet 95] [Würtz & Müller 96]
 - not of strategic importance
 - technique useful to know about

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

79

Trends and Challenges

Explanations

- Definition

A (minimal) set of constraints and choices made during search justifying a propagation event

- Uses

- Understanding dead ends
- Nogoods
- Conflict-directed backtrack search
- Debuggers and visualizers

- Challenges

- Sharp explanations for global constraints
 - Bridging semantic gap between application and CP model
- [Jussien & Barichard 00]

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

81

Meta-Programming

- Applications:

- Debuggers
- Visualizers
- Static analysers
- Search strategy synthesizers
- Test case generators
- Parser generators for propagators

- Requirement:

- Exact and formal description of all constructs

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

82

Challenges

- Communication between constraints

Constraints communicate via domain variables only, so constraints are independent of each other

- Good news: constraints can be posted regardless of already posted ones
- Bad news:
 - Loss of global view
 - Obvious propagation missing
 - Thrashing
 - Creates artificial global constraints

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

83

Challenges

- Exact and formal description of all constructs

- Syntax and options
- Declarative semantics
- Events
- Level of consistency
- Complexity

- No information should appear only in the manual

[Beldiceanu 00]

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

84

Challenges

- Hybridization
 - with local search
 - Let CP explore the neighborhood
 - with linear programming
 - Benders decomposition
- [Eremin & Wallace 01]
- Modelling languages and global constraints
- Optimization
 - Cost-based filtering algorithms
- Over-constrained problems
 - Replace $C(X)$ by $C(X, cost)$ where $cost$ is the degree to which X violates $C(X)$
- [Petit & Régim & Bessière 01]

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

85

Challenges

- Beyond finite domains
 - [Jaffar & Maher & Stuckey & Yap 94]
 - Richer set of basic constraints, e.g. TVPI
 - $X \bmod 11 \in \{1, 5\}, x \geq 2y+3, \dots$
- Classification and standardization
- Parametric constraints
 - One constraint family – one filtering algorithm
- [Beldiceanu 2000]

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

86

References

References

- [Aggoun & Beldiceanu 90]
A. Aggoun, N. Beldiceanu. *Time Stamps Techniques for the Trailing Data in Constraint Logic Programming Systems*, Actes du Séminaire 1990 de programmation en Logique, Trégastel, France, 1990.
- [Haridi & Janson 90]
S. Haridi, S. Janson. *Kemel Andorra Prolog and its Computation Model*, ICLP 1990.
- [Le Huitouze 90]
S. Le Huitouze. *A New Data Structure for Implementing Extensions to Prolog*, PLILP 1990.
- [Aggoun & Beldiceanu 91]
A. Aggoun, N. Beldiceanu. *Overview of the CHIP Compiler System*, ICLP 1991.
- [Holzbaur 92]
C. Holzbaur. *Metastructures versus Attributed Variables in the Context of Extensible Unification*, PLILP 1992.
- [Van Hentenryck & Deville & Saraswat 92]
P. Van Hentenryck, Y. Deville, V. Saraswat. *Constraint Processing in cc(FD)*. Manuscript, 1992.
- [Jaffar & Maher & Stuckey & Yap 94]
J. Jaffar, M. J. Maher, P. S. Stuckey, R. H. C. Yap. *Beyond Finite Domains*. Proc. PPCC, 1994.

CP 2002

M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

88

References

- [Puget 94]
J.-F. Puget. *A C++ Implementation of CLP*. Proc. 2nd Singapore Int. Conf. On Intelligent Systems, 1994.
- [Codognet & Codognet 95]
P. Codognet, C. Codognet. *Guarded Constructive Disjunction: Angel or Demon?*, CP 1995.
- [Puget & Leconte 95]
J.-F. Puget, M. Leconte. *Beyond the Glass Box: Constraints as Objects*. Proc. ILPS, 1995.
- [Van Hentenryck & Saraswat & Deville 95]
P. Van Hentenryck, V. Saraswat, Y. Deville. *Design, Implementation, and Evaluation of the Constraint Language cc(FD)*, Constraint Programming: Basics and Trends, LNCS 910, 1995.
- [Würtz & Müller 96]
J. Würtz, T. Müller. *Constructive Disjunction Revisited*. German AI Conf., LNAI 1137, 1996.
- [Carlsson & Ottosson & Carlson 97]
M. Carlsson, G. Ottosson, B. Carlson. *An Open-ended Finite Domain Solver*. Proc. PLILP, 1997.
- [Schulte 97]
C. Schulte. *Programming Constraint Inference Services*, CP 1997.

CP 2002 M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

89

References

- [Wallace & Novello & Schimpf 97]
M. Wallace, S. Novello, J. Schimpf. *ECLiPSe: A Platform for Constraint Logic Programming*. Tech report, IC-Parc, London, 1997.
- [Henz & Müller & Ng 99]
M. Henz, T. Müller, K.B. Ng. *Figaro: Yet Another Constraint Programming Library*. Workshop on Parallelism and Implementation Technology, ICLP, 1999.
- [Perron 99]
L. Perron. *Search Procedures and Parallelism in Constraint Programming*, CP 1999.
- [Schulte 99]
C. Schulte. *Comparing Copying and Trailing*, ICLP 1999.
- [Apt 00]
K. Apt. *The Role of Commutativity in Constraint Propagation Algorithms*, ACM TOPLAS 22(6), 2000.
- [Beldiceanu 00]
N. Beldiceanu. *Global Constraints as Graph Properties on Structured Networks of Elementary Constraints of the Same Type*. Tech. Rep. T2000-01, SICS, 2000.
- [Henz & Müller 00]
M. Henz, T. Müller. *An Overview of Finite Domain Constraint Programming*. Proc. APORS, 2000.

CP 2002 M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

90

References

- [Jussien & Barichard 00]
N. Jussien, V. Barichard. *The PaLM System: Explanation-Based Constraint Programming*. Proc. TRICS workshop of CP, 2000.
- [Laburthe 00]
F. Laburthe. *CHOCO: Implementing a CP Kernel*. Proc. TRICS workshop of CP, 2000.
- [Choi & Henz & Ng 01]
C.W. Choi, M. Henz, K.B. Ng. *Components for State Restoration in Tree Search*, CP 2001.
- [Eremin & Wallace 01]
A. Eremin, M. Wallace. *Hybrid Benders Decomposition Algorithms in Constraint Logic Programming*. Proc. CP, 2001.
- [Petit & Régim & Bessière 01]
T. Petit, J.-C. Régim, C. Bessière. *Specific Filtering Algorithms for Over-Constrained Problems*. Proc. CP, 2001.
- [Maher 02]
M. Maher. *Propagation Completeness of Reactive Constraints*. Proc. ICLP, 2002.
- [Schulte 02]
C. Schulte. *Programming Constraint Services*, LNAI 2302, 2002.

CP 2002 M. Carlsson, C. Schulte, Finite Domain Constraint Programming Systems

91